A QFD framework for quality, innovation and high-tech product development dynamics

A QFD Framework for Quality, Innovation and Hi-Tech Product Development Dynamics

Sergio ROSSETTO[1], Fiorenzo FRANCESCHINI[1], Claudio DEMARTINI[2]
(1. Department of Production Systems and Business Economics
2. Department of Computer Science
Politecnico di Torino
C.so Duca degli Abruzzi 24, 10129 Torino, Italy)

Abstract
The customer mostly chooses a product on the base of its quality, which therefore arises as the main cause of its commercial success. In a nearly axiomatic drawing, it follows that the effect of innovation is the improvement of quality, which itself becomes the aim of innovation. Even though the previous statement relates quality and innovation, it still does not explain their dynamics. To stress them, the 'quality' concept must be analyzed in more detail. In fact, in addition to the 'perceived quality', the quality ensured through `design, manufacturing and marketing' combined domains should be dealt with. This paper enhances this issue taking advantage of principles and models made available by control theory schemes coupled with quality function development (QFD) and best practice software modeling based on unified modeling language (UML).

Key words: quality function deployment; software product development; embedded systems development;
CLC number: **Document code**: **Article ID**:

Introduction

It is evident that it is not the innovative action by itself, but rather its effect on a good, that influences the customer. Any action able to augment customer judgment of the offered good is concretely innovative not only those increasing product performance, but also those improving, for example, delivery time or after-sales service, as well as those ameliorating product image. These effects, perceived and evaluated in an ordinal or cardinal manner, are transformed by the customer in a set of attributes that, together, define the `perceived quality' (Q_{pe}) of a good. The customer judges and chooses a product on the grounds of its quality, which therefore is the main cause of its commercial success. In a nearly axiomatic form, it follows that the effect of the innovation is the improvement of quality, which itself becomes the aim of innovation [5]. Even though what has been said couples quality and innovation, it still does not explain why the two functions are dynamic. To understand this, the quality concept must be analyzed more in detail.

In fact, in addition to the perceived quality, there is the quality that is actually assured by the producer through its `design, manufacturing, commercial' system. The latter is the so-called `offered quality' (Q_o). Generally, the two qualities Q_p and Q_o are different, because of the information asymmetries and the different system of weights and measures used to evaluate the product attributes. Customer evaluation is based on a reference model, founded on comparison of different products and subject to the marketing pressure of all competitors. Generally this model leads to the so-called `expected quality' (Q_e) which, for its puzzle nature, does not coincide with the Q_p of any product [6][7].

To preserve and even increase its own market share, every enterprise strengthens its effort to modify all three dimensions of quality (Q_e, Q_o, Q_p), in such a way that Q_o approaches both Q_p and Q_e. To achieve this goal, an enterprise must develop innovative actions. On the other side, since every enterprise has the same problem, all behave alike, so that Q_e and Q_p, as effects of the free market' s competition dynamics, appear as time variable functions. As a consequence, innovation cannot be an isolated intervention, but is a continuous dynamic process [9].

A first schematic representation of the innovation process is as follows. An enterprise checking a difference ΔQ between Q_e and Q_p develops two complex actions to increase customer judgment of its product: the first, through marketing, is directed to conditioning the customer; the second, through a series of technical-organizational interventions, is focused on improving the designing, manufacturing, supporting system so as to obtain an intrinsically superior product. The two actions are always present, although the relative intensity depends on the market sector, the maturity of the product and the customers' culture. In any case, in order to intensify the results, both actions must be coordinated by means of an adequate systemic approach in enterprise management.

From a control-science point of view, the innovation process can be synthesized in two distinct feedback loops: the first with a prevalent `communicative-persuasive' content, the second with a prevalent

`engineering-organizational' character. The `communicative-persuasive' channel, managed by the marketing function, has the target of modifying Q_p, and of inserting in Q_e some peculiar attributes of the offered product. The main aim of the engineering-organizational channel, on the contrary, is to improve Q_o. If the conceptual model has some appeal for its theoretical use, two main problems have to be solved. The first one concerns the construction and the identification of:

$$I(\delta Q_o/\delta t, Q_o, PI) = 0 \text{ and } J(\delta Q_p/\delta t, Q_p, MI) = 0 \qquad (1)$$

where PI and MI are production and marketing interventions, and $\delta Q_o/\delta t$, $\delta Q_p/\delta t$ the first derivatives in time of the offered and perceived quality. The second problem regards both the evaluation and the comparison of Q_e, Q_p and Q_o; for the last point, multi-criteria decision analysis techniques seem to be particularly suitable [3].
In order to take theoretical aspects into account together with qualitative issues, quality function deployment (QFD) [1][2][8] is recalled in section one, analyzed according to the ICT perspective in section two; connections with UML features and tools are described in section three, while issues on the `engineering-organizational' scenario taking advantage of an action-feedback-correction schema are dealt with in section three. This approach provides a model to identify solutions for (1), as section four describes in detail, where the QFD framework plays a relevant role for building both control and regulation mechanisms.

1. Quality function deployment and customer requirements

QFD is a method for product/service planning and development that enables design teams to specify customer's needs supporting evaluation of each proposed product or service capability by processing data gathered to describe its impact with respect to those needs [4]. It was born in Japan during the 1960s and was planned to develop a new product meeting customer needs and providing quality control process charts to manufacturing before production starts. QFD provides a structured concurrent framework to translate customer requirements into corresponding technical ones drawing subsequently component characteristics, process and operational steps. Each translation uses a complex matrix usually defined HoQ, as shown in Figure 1 [4]. The HoQ process is often summarized into the following steps:
1. Defining customer attributes (CAs); 2. Assigning weightings to CAs; 3. Conducting customer competitive analysis; 4. Producing engineering characteristics (ECs) in response to CAs; 5. Studying technical correlations between ECs; 6. Determining relationships between ECs and CAs 7. Processing importance weightings of ECs; 8. Setting target values of ECs; 9. Weighting technical competitiveness of ECs.

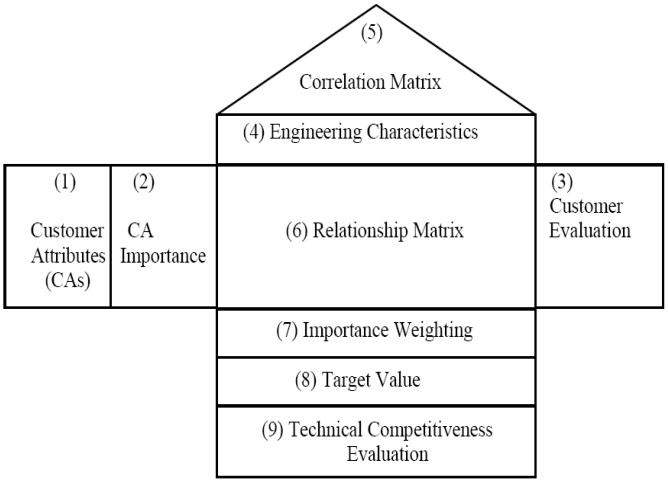

Fig. 1 The House of Quality (adapted from Cohen)

The applications of QFD have reported a variety of benefits to many organizations. During this period, the product development cycle was reduced by one third with a corresponding increase in quality because of a reduction of the number of engineering changes.
However, the practical implementations of QFD in the various contexts of industry have encountered difficulties and, in some cases, even strong oppositions, as happened in the software development domain. In any case, although design-relevant software process knowledge and information are usually fuzzy, incomplete, and even conflicting in nature, the conventional QFD applications, always assuming that the

input data are precise, force the QFD team to build consensus views, overcoming entropy and divergence with respect to user requirements collection and classification.
It is also difficult for QFD to support the process of software development in which design teams need firstly to satisfy client requirements and then use at best the software knowledge and expertise to achieve an improved design with a suitable or increased client satisfaction. Furthermore they also have to assess the degree of satisfaction of client and software implementation needs respectively. In particular, in the field there is lack of a supportive environment to successfully apply QFD due to the tight interlacement of design and development of modern software process life cycle.
Hence, in principle, QFD in its classic form can be used to develop software products [10] even if at least two essential differences must be taken into account when applying QFD to software development. In the software industry the production process in a strict sense is a mere duplication process, just as the definite implementation process cannot be influenced by special adjustable parameters. Therefore, the problem for the software scenario lies in a still higher degree than in manufacturing in the early phases of development. Applying QFD to software development, therefore, has to focus on the ability to prioritize the engineering activities and pay less attention to deployment down to the software's last line of code [10].

2. QFD and the software development process

QFD is gaining attention in software development contexts, in fact it has been proposed as a means to formalize collection and translation of customer needs into a set of system design features since software complexity has began to overcome sustainable effort according to the offered performance of the traditional development tools available. Hence, Software QFD (SQFD) was introduced to provide formal approaches to link high-level customer requirements to specific system requirements [11][13]. Various global corporation active in software and complex system development reported employing QFD in software development such as IBM, SAP, Siemens, Toshiba, Texas Instruments, and Hughes Aircraft [20].
Most of improvement comes from Zultner's model for applying QFD to software development, being it extensive, detailed and based on the use of several basic matrices [15].
In this scenario the intangible nature of software does not help customers, who often find it difficult to deal with quality when expressing requirements.
A major advantage achieved implementing QFD include the capability to orient the discussion in order to deal with the kernel of the problem instead of working on peripheral problems involving statements on what customer requirements are with respect to what design is in the context of the scene to be analyzed. Relevant issues on which the tool focuses are:

- Identification and classification of customers according to the market segment they belong also paying attention to their different and specific needs;
- Weighting customer needs in order to build a representative requirements space as a framework into which exploiting measurable quality issues;
- Differentiation and clear identification of product and service features that customer requires
- Sustainability of the correlation among product/service properties and the pooled wishes declared by the customer.

Richardson and Ryan in [17] have developed a generic method based on QFD that can be used by small software development companies in the derivation of their software process improvement strategies. The authors implemented the method, the software process matrix (SPM), in two small software development companies in Limerick, Ireland. The SPM is based on the first matrix of this model, the HOQ which can be applied to software design, and the resulting software requirements are diverse in their scope and coverage. The result takes the shape of a product acceptance extended beyond basic functionalities to serve as an indicator of reliability, usability, and other customer preferences and design considerations. This contributes to face the problem that in software, 50 to 60 percent of defects originate in the requirements phase. Following the line, QFD showed to be a proven technique able to reduce the number of defects, subsequently resulting in gains for product development and customer satisfaction.
In this cases a slightly different approach was proposed, compared to that one introduced by Zultner's. In fact despite the fact that SQFD must focus on early phases of development, Zultner proposed a framework centered on how to apply Akao's comprehensive QFD to software development, including quality deployment according to the four phase scheme. The most important additional aspect is the emphasis on customer deployment treated before quality deployment takes place, this approach is derived from the feeling that software, as happens with most other products, rarely is conceived to satisfy the needs of only one

homogeneous customer. To identify customer groups and their importance for the engineering ahead, a table showing potentially relevant customer characteristics and prioritization matrix of selected criteria is used.
Quality deployment as a second step combines classic and software HoQ to form one single HoQ, so that functional as well as non functional product characteristics are being considered in line with Akao's framework. It is through information deployment that at least the first two phases of the four-phase model may be easily mapped on software development cycle. In information deployment the product characteristics that have been prioritized in the HoQ are turned into entities, relationships, attributes, processes or objects, depending on the development techniques used. Furthermore, Zultner's function deployment still proceeds similarly also when hardware design is required and task deployment is concerned with activities of the development process itself. Vertical deployments are intended to guarantee that general aspects such as reliability and costs are taken into account in any development activity. In these contexts measurable quality features taken from the HoQ are particularly important.

3. A link between QFD and UML

In contrast with this comprehensive QFD framework, Zultner himself reduced his QFD approach to absolutely essential activities [16]. This model called 'Blitz QFD', focuses entirely on the reception, analysis and weighting of customer requirements, after customers have been identified (customer deployment) and before the HoQ has been set up (quality deployment). This simplified model has been assumed as the starting point in this work to develop a smart connection to UML diagrams able to provide full support to information deployment within the whole software development cycle.
In order to take into account the various perspectives according to which different user experiences take place [14], two main categories are introduced to classify persons and organizations. In both the cases managers, operators, developers, maintainers and end users are often considered to be relevant counterparts to get specific sets of requirements representing those roles perspectives. Fig. 2 draws the HoQ translating user requirements clustered, according to the previous rationale, into engineering characteristics, which in turn are also classified according to functional and non-functional super classes further subdivided into Hardware, Software and Infrastructure classes. This approach makes the link between QFD and UML possible and easier, given the opportunity offered by QFD to identify any single functionality to be placed in correspondence with a Use Case. This link enforces the capability to describe technical features in terms of detailed specifications extended beyond pure user requirement reinterpretation and taking advantage of the whole suite of UML diagrams [11][12].
Another issue promoted by the model proposed in this work concerns a wider classification of both user requirements and engineering characteristics, which are described on the base of properties which allow analysts to distribute them into the following categories: ergonomics, economy, ethic, capability and dependability [18].

3.1 Ergonomics

Ergonomics, also defined 'human factors', concerns the understanding of interactions among humans and devices, being them often part of a wider system.
In particular 'physical ergonomics' deals with human properties, biological and mechanical characteristics as they relate to activities addressing working postures, objects handling, repetitive single translations of parts of the body, work connected disorders, layout, health and safety. On the other side of the coin there is 'Cognitive ergonomics' which is concerned with perception, memory, reasoning, motor response, affecting action-reaction among human actors and other components of a system. In this domain relevant topics include process decision, skilled performance, human-computer interface, socio-technical system reliability, work overload and training as these may be linked to social systems design and development [19].

3.2 Economy

Being economics the social science that studies production, distribution, and use of goods and services, in the following it is intended to limit its semantics only to a category useful to gather issues relevant for costs of goods and services. To this purpose attention is mainly paid to equipment operating costs, since our interest is mainly focused on product/service end-user, including maintainers and operators rather than developers, leaving all the rest to future debate outside the perimeter of this work. In the case of equipment, that is a device, a component or facility, they are the usual and customary recurring costs of operating the equipment itself, costs which arise regularly in time, and do not include the capital cost of constructing or purchasing it. Operating costs are incurred by all equipment, unless they have no cost to operate, require no person to be run

or applied or no space as it may happen with intangibles like software modules or applications which, in some cases, may even appear to have low or no operating cost because either the cost is not identified or is being absorbed in whole or part by that charged on the hosting hardware. Hence equipment operating costs may include: salaries of human resources, advertising, components and raw materials, license, registration fees, real estate expenses such as rent or lease, office spaces and related furniture and equipment, mortgage of funds used to buy land for ownership, property taxes, operation taxes, transport fees, fuel costs for operation and production, public utilities, maintenance, office consumable, insurance, income taxes, and so forth [18].

3.3 Ethic

In this framework many European countries, such as the Netherlands, Germany and other northern countries, promote environmental issues and even regulate them and underpinned by laws. The paradigm of the product fitting everyone took shape through the rise of the design for all concept aiming at developing products that would be usable by as many people as possible, being them often part of the aging population domain. However the interest has recently shifted from this to a wider perspective on accessibility since the frustration derived from packaging complexity which makes products difficult to be accessed or products too complicated to be intuitively understood by users independent of their age or capability. In any case two paradigms co-exist now in total conflict with each other: the sustainability model, describing the world as a system of ecological checks and balances based on finite resources, and the expansion model, which represents the world in terms of markets where products play the role of tokens of economic change [22].

3.4 Capabilities

Functional requirements express the intended behavior of the system which may be described referring to services, tasks or functions the system is required to perform according to user needs and specific objectives defined. Product development is carried out distinguishing between baseline functionalities necessary for any system to compete in that product domain, and other features that differentiate the system from competitors' products, and from variants already made available within the company product family. A strategy to penetrate the market, even for software products, is to produce the core of the basic product, adding features to its variants and release them shortly thereafter. In many sectors, companies produce product lines with different cost/feature variations per product, collecting them in families targeted at different market segments or application types, identifying specific product lines having some common elements which specify their functionality and better represent their identity. Software architecture are leveraged by these strategies since it is not just the functional requirements of the first product release that must be supported by the architecture, but also later releases should be considered and adjusted through added architectural properties such as flexibility, extensibility, performance and so forth, being them mostly non-functional requirements. In this context use cases provided by UML have quickly become a widespread practice for capturing functional requirements within the object-oriented community where they originated [21].

3.5 Dependability

Dependability ensures that *availability*, *reliability*, *security* and *safety* targets be achieved before a new product goes into full-scale production during product development and pilot production phases. When planning new product features and capabilities, issues concerning risk are inevitably to be dealt with. Risk assessment and mitigation take place on the base of systematic measures and comparisons among alternative solutions carried out early in the project life cycle. This implies investment in planning, analysis, specification, engineering and validation reducing warranty costs, enhancing quality and enforcing reliability also managing other expenses associated with infringed dependability. The aim is to pursue the overall system reliability through specific engineering activities carried out to frame the best prototype dependability level, through systematic identification and resolution of problems which might have not been anticipated and correctly estimated in earlier design stages. This is particular true for critical systems for which the most important system property is effectively the dependability of the system which reflects the user's degree of trust in that system [18].

4. QFD Ontology and the closed-loop feedback control

The diagram reported in Fig. 2 describes a part of a data base structure planned to collect and maintain information related to the closed-loop control put in place to follow the market response with respect to the product/service made available to consumers at the end of the development process. The following drawing gives evidence to control system where a sensor monitors the output, as happens for example with a vehicle

speed, and provides data to a computer which continuously adjusts the control input, just as behaves the throttle when it is repositioned as required to keep the control error to a minimum in order to maintain the desired speed. Feedback on how the system, whatever it may be, is actually performing allows the controller, as happens with a vehicle's on-board computer, to dynamically compensate for disturbances applied to the system itself. For example disturbances may be expressed by changes in slope of a road or even the wind speed pushing or delaying the running vehicle.

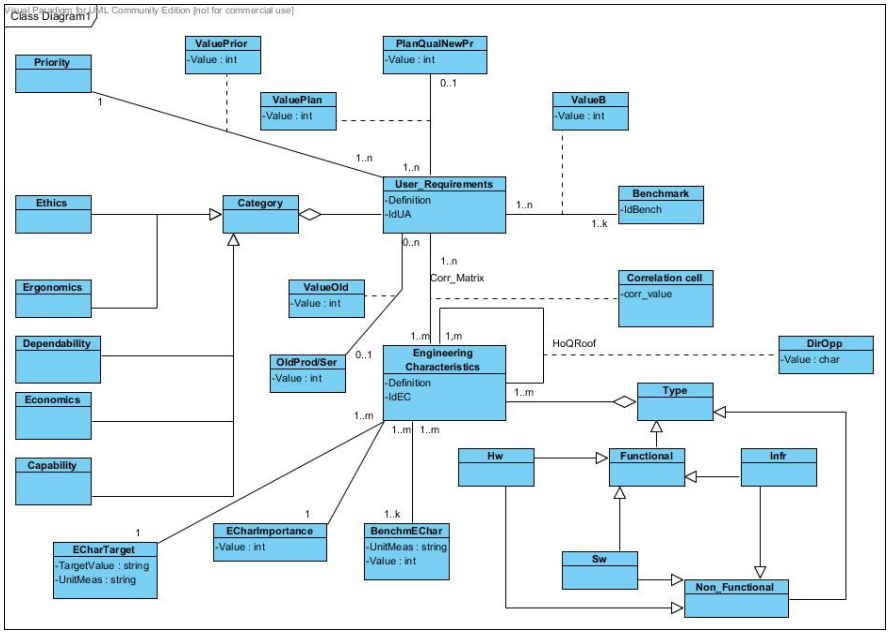

Fig. 3 The ontology representation

It is expected that an ideal feedback control system can compensate completely any error, balancing the effects of any force that might arise during system operation, providing a response that perfectly matches the system reference or device target. Of course this is an unachievable scenario due to measurement errors occurring in the sensors, delays experienced in the controller, and also to imperfections in the input fostering the controller itself.

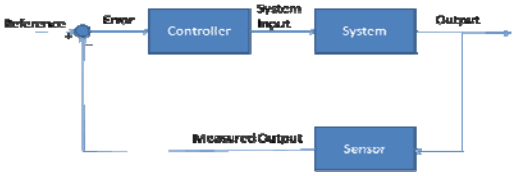

Fig. 3 Closed loop control

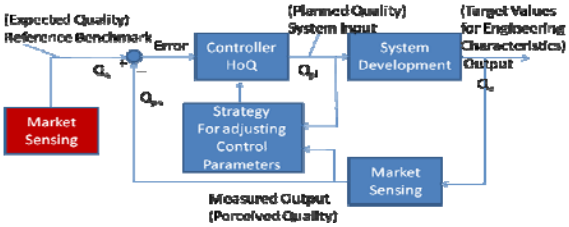

Fig. 4 Closed-Loop Control and HoQ

The general model reported in Fig. 3 is then updated in order to make it apt for the product development scenario which has been drawn in the previous paragraphs. The new diagram in Fig. 4 is in fact built enhancing the previous one by adding an adaptation block in order to keep track of market dynamics. It is still organized as a closed-loop schema, but acting as an adaptive control, since the controller itself can change its own configuration parameters in order to follow updated market response.
The diagram includes the following main blocks:

- Controller: it is composed by the whole HoQ composed of the user requirements and the product/service characteristics vectors, together with the relationship matrix and the product features importance; benchmark evaluation vectors are also included;
- System development: which represents the product/service development processes and includes product development life cycle and communication plans;

- Market sensing: it is based on solutions adopted to analyze the market, often based on real time web based solutions to get opinions and wishes from segmented user domains of the product/service;
- Strategy for adjusting control: it senses the market to update the HoQ relationships matrix values and also to shape requirements and product features vectors.

The reference signal is obtained by sensing the market with respect to benchmark data collected gathering information provided by customers who use the best competing products in the market described according to the customer requirement space CR_i ($1 \leq i \leq n$), whose dimensions are shaped by all identified requirements. The same dimensions, that are the customer requirements, are those used to structure the house of quality able to work as a controller within the regulation loop. Sensing the market, besides customer opinion expressed through requirements, also brings to light specific engineering characteristics of a product/service, relevant issue for being able to specify the controller shape and response profile when solicited by the error signal. The latter is computed in terms of difference between expected and perceived quality, both expressed within the same customer requirements space already defined for benchmarks representations. The first suggestion coming from the model is that dimensions of customer requirements and engineering characteristics (EC_j, $1 \leq j \leq m$) are always provided by the market whenever appropriately sensed. The adjusting control block gets sampled data to shape jointly both CR and EC, assigning relationships value for any requirement-characteristic correspondence and specific relative weights to any CR.

Estimating EC weights is the next step, being subsequently used to determine which particular technical requirements (EC) are important to improve first, so that effort could be concentrated on them for quality enhancement. In other words the model states that the quality plan (Q_{pl}) fosters the 'System Development' block, and that plan must be composed of a set of vectors including 'Importance Weighting', 'Target Values of EC' together with the one representing the 'planning phase' of the company in terms of associated quality. The whole input is used as information to carry out all other phases which, in software development domain, are based for the most part on UML diagrams devoted to software modules description and supporting semi-automatic coding process. It is interesting to stress the role of the sensor, which in this specific case, is framed as customers, suitably segmented, and questionnaires used 'to sense' them in order to measure the "perceived quality" (Q_{pe}) to be compared to that expected (Q_e) in order to generate the error signal operating as a driver for the controller. It is evident in this scenario that the special sensor introduced in this process control is highly affected by various kind of disturbances, which can cause a relevant distortion of the sampled signal, the perceived quality, which is of course represented within the same space defined for customer evaluation and already used to sample benchmarks and planning phase within the loop. It is interesting to emphasize the fact that quality perceived is built through the use of proper questionnaire shaped on the customer requirements to be analyzed (Q_{pe}), and the opinion customers express are formed on the real products they handle, use, apply and live on a day by day basis, hence those assessment are in most of cases derived directly from a physical experience shaped on the effective features of those product or services.

The 'System Development' block includes main operating domains basically founded on: 'process', 'communication' and 'technology', being them the three macro areas within which Q_{pl} can be spent. The expected output of this block is twofold, the real product, on one side, whose properties should perfectly match the specification given at the block input and laid out in terms of 'Target Values of EC', having the same descriptor accompanying the block output, on the other side there is the communication plan established to make the product/service virtually captured by the market segments potentially interested in it. To bring to light the new role and relevance of technologies, social media should be mentioned, which has become a platform that is available to anyone provided with internet access, opening doors for organizations to increase their brand awareness and facilitate interaction with the customer. Additionally, social media serves as a relatively inexpensive platform for organizations to implement marketing analysis where direct feedback from customers and targeted markets can be efficiently processed.

5. Conclusions

Knowing that the effect of innovation is the improvement of quality, which itself becomes the aim of innovation, this work focuses on their dynamics. To stress them, the 'quality' concept is analyzed in more detail arguing on 'perceived quality', that ensured through `design, manufacturing and marketing' combined domains and promotes solutions able to compare it with the expected quality taking advantage of principles and models made available by control theory schemes coupled with quality function development and best practice software modeling based on unified modeling language. To define the whole schema, database and software engineering techniques have been used in order to reduce the time required to extract information

got sensing the market and necessary to sustain the closed loop feedback control, the latter implemented according to an adaptive approach.

References

[1] Akao V. (1988), Quality Function Deployment, Productivity Press, Cambridge (MA).

[2] Chan, L.-W. and M.-L.Wu., (2002), "Quality function deployment - A literature review.", European Journal of Operational Research, v. 143, pp. 463–497.

[3] Roy, B. (1991) The outranking approach and the foundation of ELECTRE methods, Theory and Decision, 31, pp. 49 - 73.

[4] Cohen, L., (1995) "Quality Function Deployment", Addison-Wesley, Reading.

[5] Franceschini, F. and Rossetto, S. (1995), ``Quality & innovation: a conceptual model of their interaction", Total Quality Management, Vol. 6 No. 3, pp. 221-229.

[6] Franceschini F., Rossetto S. (1995), "QFD: The Problem of Comparing Technical Design Requirements, «Research in Engineering Design», v. 7, pp. 270-278.

[7] Franceschini F., Rossetto S., (1997), Design for Quality: Selecting Product's Technical Features, «Quality Engineering», v. 9, n. 4, pp. 681-688.

[8] Franceschini F. (2002), Advanced Quality Function Deployment, St. Lucie Press/CRC Press LLC, Boca Raton, FL, USA.

[9] Franceschini, F., Galetto, M., & Maisano, D. (2007). Management by measurement: Designing key indicators and performance measurement systems. Springer Verlag , Berlin.

[10] Helferich A., Herzwurm, G.; Schockert, S, (2005) "QFD-PPP: Product Line Portflio Planning Using Quality Function Deployment", Lecture Notes in Computer Science, v. 3714/2005, pp.162-173, Springer Berlin, DOI: 10.1007/11554844.

[11] Herzwurm, G., Schockert, S., Mellis W. (2000) "Joint requirements engineering: QFD for rapid customer-focused software and Internet-development, Vieweg, Wiesbaden, Germany.

[12] Herzwurm. G., Schockert, S., Friebel S. (2004) "Quality Function Deployment objectoriented – a method for the combination of Quality Function Deployment and object-oriented modeling". In: Proceedings of the 10Th International Symposium on QFD, Monterrey, Mexico.

[13] Nuseibeh, B., Easterbrook S., (2000) "Requirements Engineering: A Roadmap". Proc. of the Conference on The Future of Software Engineering, Limerick, Ireland (2000), pp. 35 – 46.

[14] Georg Herzwurm, Sixten Schockert, Wolfram Pietsch: QFD for Customer-Focused Requirements Engineering. RE 2003: 330-345.

[15] Zultner, R.E. (1990), "Software Quality [Function] Deployment. Applying QFD to software", in QFD Institute (Eds),Transactions from the 2nd Symposium on Quality Function Deployment, QFD Institute, Novi, MI, pp.132-49.

[16] Zultner, R.E. (1995), "Blitz QFD: better, faster, and cheaper forms of QFD", American Programmer, pp.24-36.

[17] Ita Richardson, Eamonn Murphy, KevinRyan, "Development of Generic Quality Function Deployment Matrix", Quality Management Journal, Vol. 9, No. 2, April 2002, pp. 25--43.

[18] Ian Sommerville, Software engineering 9, March 2010, Pearson Education.

[19] Kathryn Cormican; David O'Sullivan (2007) A groupware system for virtual product innovation management, Human Factors and Ergonomics in Manufacturing, 17(6), p499-510, Wiley InterScience.

[20] Cristiano, J.J.; Liker, J.K.; White, C.C. III., (2001), Key factors in the successful application of quality function deployment (QFD), Engineering Management, IEEE Transactions on, 48(1), p81-95.

[21] Wei Xiong; Xiao-Tun Wang, (2008), Software Requirements Management using QFD: A Process Perspective, Computational Intelligence and Design, 2, p295-299.

[22] Margolin, Victor (2002): The Politics of the Artificial. Essays on Design and Design Studies, The University of Chicago Press, Chicago.